Victorian 6502 User Group Newsletter

KAOS

For People Who Have Got Smart

OSI SYM KIM AIM UK101 RABBLE 65

Registered by Australia Post
Publication No. VBG4212

ADDRESS ALL CORRESPONDENCE TO 10 FORBES ST., ESSENDON, VIC. 3040

Vol.5 No.2 November 1984

We have some good news and some bad news for you this month. The bad news is that membership fees are due soon. Your renewal membership form is enclosed in this month's newsletter. The good news is that fees will not be increasing, if you can't remember how much you paid, it's on the form.

On the subject of good news; the next meeting will be a family get-together and we will be having a BBQ in the school grounds. We will supply the cooking facilities, you supply the rest. The school grounds will be open at 11am.

The next meeting will be on Sunday 25th November at 2pm. at the Essendon Primary School which is on the corner of Raleigh and Nicholson Streets, Essendon.
The closing date for articles for the December newsletter is 7th of December.

FOR SALE

C1P with Tasan video board (64x32 screen), MPI 51 disk drive, C4 Dabug, parallel printer interface, RS232C interface, 32K RAM and Bernie Wills' enhanced character set. Software includes WP65Ø2 word processor and some games. This machine can be used as a disk based system or a cassette based one.

I expect to be travelling East during January so I will be able to personally deliver this machine to most points between Perth, Adelaide, Melbourne and Canberra. Cost $55Ø ono.
Gerry Ligermoet,

BRAND NEW OSI COMPUTERS:
1 only C1P Series 2 cassette based for $25Ø
1 only C1PMF Series 1 (metal case) for $54Ø
1 only C1pMF Series 2 (plastic case) for $56Ø
Various program cassettes offer
8 Superboards ratted for spare keys etc. $5Ø ea.
Filip Coelho,

INDEX

C1P DOS NOTES
by Mark Howell

1. I've found two small errors in my copies of CompDOS 1.2, at $2788 code should be A9 2Ø not 89 2Ø and at $2969 code should be 85 F7 not 85 B7.

2. There are two areas of apparently unused code in 1.2. They are $7DED - $7DFF and $7FE5 - $7FFF.

3. A fix for the "CREATE" command bug (see V3/£8 P12) is to alter code at $7C37 to 98 F8 38 E5 DE 4C F6 7D or 4C F6 7F and at either $7DF6 or $7FF6 add C5 DD BØ Ø3 4C 3F 7C 4C 48 7C.

4. The "SWAP 4" routine in 1.2 is at $25ØD not $2644 which hasn't been corrected in the "RE MON" command around $2C19. Code should be 2Ø ØD 25.

5. Bug in original code at $7D4E. (See V3/£6 P6) FØ Ø6 should be FØ Ø4.

6. The "Q" command in the Extended Monitor needs to be altered for the 48x12 mode. Change code at $18D9 from A9 15 to A9 ØB.

7. The message code at $29FD seems to be incorrect. On OS65D3 it is ØD ØA 52 41 43 4B 2Ø.

8. The "SINGLE KEY" table begins at $7FA1. Change these locations to alter the BASIC command used for each key.

9. For anyone with HEXDOS 4 and a SYN6ØØ or DABUG monitor a POKE to 2329 will alter the SHIFT RETURN used in the line editor. For example, to use SHIFT L. POKE 2329,92.

KAOS NSW
by Norman Bate

The October meeting saw the attendance of 8 members and 3 computers. Nigel Bisset had his new Apple Macintosh and Julian Arul had a borrowed Porta Pak. The obvious hit was the Mac judging by the queue waiting to draw and paint and print-out using the different fonts.

As previously advised the December meeting is on Sunday the 16th. However, to cater for those living in the west, the meeting will be at member Brian Martin's house,

His phone number is

All members should make an attempt to be there for some time during the day. Those bringing computers will have to bring a table for their gear.

For information contact N. Bate or N. Bisset

KAOS W.A. in CHAOS

In the last newsletter I indicated that we would be having a meeting on Sunday 16/12/84 at Peter Van der Wedden's home. Unfortunately due to a mixup on my part this meeting will have to be cancelled as Peter will be overseas at the time of the meeting.
Gerry Ligtermoet

THE CP/M FILE
by Ralph Hess

It looks as though the call put out last month has been heard, in some sectors at least. At the time of writing, we have ten interest group members. If there is anybody else out there who wants a piece of the action, please get in touch with me. This applies to country and interstate KAOS members as well. The only extra cost is that of communicating.

The October KAOS meeting saw the official (?) establishment of our group. The pippen brigade decided that that seemed like a good idea and promptly copied it.

Those of you who have K83-BC keyboards (IBM style with 1Ø function keys on the left) may be interested to know that we have a modified ROM available for it to suit CP/M and Wordstar and dBaseII. Further improvements, with even more packed into the ROM, are being worked on, and we hope to end up being able to adapt it to other keyboards that use the 8Ø35/8Ø39/8Ø48 CPU.

On top of all that your correspondent has it on good (?) authority that things are happening insofar as operating systems are concerned. With a little bit of luck, a lot of sweat and many muttered magic spells by one P.G.D. (et al.), we could soon have our systems turbo-charged.

Also planned for 1985 (that's as close as I can get) is an I/O port expansion card with 15 user ports, a real-time clock and priority interruptfor about ten of the ports. This will replace the 4 I/O ports on the Rabble and just plug into the 4Ø-pin expansion bus on either OSI or Rabble.
That's all for now, folks. See you all at the BBQ.

TAPE TOKEN FORMAT FAULT
by John Whitehead

I found that sometimes I could not load in a Token format tape made with the program I published in KAOS Oct. 83, when Dabug was off. The BASIC leader would load, but the M/Code would not.
After some detective work, I found:-
PKOE11,67:POKE12,254:X=USR(X):END
causes a jump to the monitor at $FE43. (67=$43, 254=$FE). The code at $FE43 prints AAAA@DD on the screen, then branches to $FE2A, jumps to sub at $FE39 which checks memory $ØØFB to see if it contains $ØØ. If it does, it waits for a keypress, if not, it waits for input from the ACIA (tape).

The problem is that on a series II, memory $ØØFB (decimal 251) is also used for screen size, and may still contain $ØØ with Dabug off in 24x24. This will cause the Token program on the the tape being loaded to wait for a keypress, in place of getting more data from the tape.

To fix the fault in the Token program, add line:-
2Ø2 PRINT" 27 IFPEEK(251)=ØTHENPOKE251,2

If you have trouble loading library tapes I have made, press L after BASIC runs so that the M/Code section gets loaded.

I have also altered the Token program so that I can save BASIC and M/Code together and the M/Code can be fetched from a different area to that in which it will load. Contact me for details.

SUPERBOARD

Newsletter of the Ohio Superboard User Group, 146 York Street, Nundah 4012

FASTSORT FOR OSI

Fastsort is the last of the "fast" series. We had Fastdraw (followed by fastdraw revisited), Fastsearch, and now Fastsort. Fastsort is by far the most complex of the three, and despite many hours of pondering and disassembly, I still don't fully comprehend it.

Fastsort was written to work with Basic-in-Rom, by Alan Cashin.

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
1C00  A2 08 BD 95 1D 95 13 CA D0 F8 86 5F A0 05 B1 AA
1C10  C8 91 1A B1 AA C8 91 1A 98 18 65 AA 85 1C 8A 65
1C20  AB 85 1D 20 29 1C 4C 23 1C A0 00 B1 18 29 0F F0
1C30  02 85 2C B1 18 29 F0 4A 4A 4A AA BD 80 1D 48 85
1C40  1F BD 81 1D 48 85 1E B1 1E F0 0D AA A8 B1 18 F0
1C50  03 99 2C 00 88 D0 F6 8A 38 65 18 85 18 98 65 19
1C60  85 19 60 01 A5 2C 25 13 F0 F8 18 A5 2D 10 EA 88
1C70  30 E7 00 68 68 60 01 20 2B 1D A0 01 B1 20 91 22
1C80  88 10 F9 60 01 20 2B 1D A0 01 18 B1 22 71 20 91
1C90  22 88 10 F7 60 01 20 2B 1D A2 02 A0 00 B1 20 D1
1CA0  22 4C 90 1E B1 20 D1 22 F0 05 CA 90 02 A2 04 86
1CB0  13 60 00 20 47 1D A0 03 B1 24 99 14 00 88 10 F8
1CC0  60 00 20 47 1D A0 00 A2 02 A5 14 D1 24 F0 07 CA
1CD0  90 04 A2 04 B1 24 85 1E 88 C8 C4 1E F0 0C B1 28
1CE0  D1 15 F0 F5 A2 01 B0 02 A2 04 86 13 60 01 20 42
1CF0  1D A0 03 B1 24 AA B1 26 91 24 8A 91 26 88 10 F3
1D00  60 00 20 30 1D A0 00 B1 20 4A 91 20 C8 B1 20 6A
1D10  91 20 60 01 20 30 1D A2 00 F0 06 01 20 30 1D A2
1D20  01 A0 01 18 B1 20 65 2D 95 2C 60 A2 01 20 32 1D
1D30  A2 00 4C F0 1D 98 0A 65 1A 95 20 A9 00 65 1B 95
1D40  21 60 A2 01 20 49 1D A2 00 20 32 1D B5 20 85 1E
1D50  B5 21 85 1F A0 01 B1 1E 95 24 88 B1 1E 16 24 2A
1D60  16 24 2A A8 B5 24 65 1C 95 24 85 1E 98 65 1D 95
1D70  25 85 1F A0 02 B1 1E 95 29 88 B1 1E 95 28 60 FF
1D80  1C 72 1C 76 1C 84 1C 95 1C B2 1C C1 1C ED 1D 01
1D90  1D 13 1D 1B 1C 63 00 00 00 00 00 1E 9C 1D FF FF
1DA0  00 01 00 14 00 00 FF FF 00 08 00 07 00 09 00 07
1DB0  00 10 00 11 00 08 00 01 00 11 00 13 00 EA 01 0B
1DC0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1DD0  FF FF FF FF 00 09 00 03 00 13 00 14 00 EC 01 0C
1DE0  FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
1DF0  B4 2C 8A 0A AA 4C 35 1D FF FF FF FF FF FF FF FF
1E00  11 08 14 09 59 A2 08 22 08 38 09 A2 02 68 09 22
1E10  09 39 03 A1 EF 18 03 33 04 A2 70 33 01 A2 6C 14
1E20  05 14 0C 13 1C 85 0C 10 06 85 1C 10 07 16 08 17
1E30  09 16 0A 27 0A 7A 4A 58 A3 04 22 08 A7 F9 59 A6
1E40  04 21 09 A7 F9 38 09 A4 0A 68 09 22 08 21 09 38
1E50  09 A3 E4 18 0A 29 0A 16 0B 27 0B 3A 0B A6 12 38
1E60  07 A6 0A 95 0C 18 00 95 1C 17 00 22 05 19 07 A7
1E70  10 36 09 A6 0A 95 0C 16 00 95 1C 19 00 22 05 18
1E80  06 36 07 A1 A8 21 05 35 01 A5 9A 00 FF FF FF FF
1E90  F0 0D B1 20 51 22 10 04 2A 49 01 6A 4C AA 1C C8
1EA0  4C A4 1C FF FF
```

The program sorts string arrays. For practical use, you would read in all the strings from data statements or off cassette, and they would then be sorted into alphabetical order and printed out. Doubtless, the program could be modified to work with Disk Basic and/or disk files, but I leave the conversion for someone more interested in that area. The following program is a short driver routine to exercise the Fastsort. It will sort a hundred strings of any length in less than two seconds.

```
10 POKE 11,Ø:POKE 12,28:CLEAR:DIM A$(5ØØ)
15 INPUT"Number of strings";N:INPUT"Length of strings";L
20 FORR=1TON:FORP=1TOL:A$(R)=A$(R)+CHR$(65+INT(26*RND(1))):NEXT
25 PRINTA$(R)" ";:NEXT:PRINT:PRINT"--Sorting--":R=USR(A$(Ø))
30 PRINT:FORR=ØTON:PRINTA$(R)" ";:NEXT:END
```

ANOTHER BASIC 3 GARBAGE COLLECTOR FIX

Early in 1981, OSUG started offering this Basic 3 fix, after we tried the Aardvark Basic 3 and found it considerably less than perfect. We haven't heard of any problems with this modified eprom, and it offers a much faster garbage collection than the one published by Earl Morris in KAOS. If you are programming your own eprom, this offers a much simpler change.

OLD CODE		CHANGES FOR GC FIX	
BIB7 B1 71	LDA ($71),Y		
B1B9 ØA	ASL A		
B1BA 69 Ø5	ADC #$Ø5		
B1BC 65 71	ADC $71	B1BC 2Ø ØB B2	JSR $B2ØB
B1BE 85 71	STA $71	B1BF E6 AØ	INC $AØ
B1CØ 9Ø Ø2	BCC $B1C4	B1C1 EA	NOP
B1C2 E6 72	INC $72	B1C2 EA EA	NOP NOP
B1C4 A6 72	LDX $72	B1C4 EA EA	NOP NOP
B1C6 E4 A5	CPX $A5		
B1C8 DØ Ø4	BNE $B1CE		
B1CA C5 A4	CMP $A4		
B1CC FØ C3	BEQ $B191	B1CC FØ BB	BEQ $B189
B1CE 2Ø D9 B1	JSR $B1D9		
B21C A5 A2	LDA $A2		
B21E 29 Ø4	AND #$Ø4	B21E 29 Ø2	AND #$Ø2
B22Ø 4A	LSR A	B22Ø 18	CLC
B221 A8	TAY		

The corrected Eprom is offered for $9.80 incl postage, or for $21.60 with our Basic 4 Eprom with high speed cassette LOAD/SAVE/VERIFY routines for both Basic and M/C programs. Specify if your cassette port is at $FCØØ or $FØØØ if ordering Basic 4. Basic 4 ordered separately is $12.80.

MISCELLANEOUS BITS

Alan Calvert bought some software from Premier in the UK, and the catalog they sent indicates that they are destocking their range of UK1Ø1 and OSI expansion PCBs , Eproms and etc.

Because of problems in getting free Sundays, in future I will advertise coming KAOS Queensland meetings the week before in the computer section of the Weekend Shopper columns.

NEXT MONTH Another program review. Part 2 of the Adventuring series.

UNDERSTANDING FORTH part 5

by Ray. Gardiner

5.INTERPRETER

The forth system really comprises two interpreters one is the inner address interpreter, the other the text interpreter defined by the word INTERPRET our aim this month is to attack the address interpreter in some detail, and show how the address interpreter operates, to show this in detail we need to look at the dictionary structure.(more on dictionary structures in part 8).

Each entry in the forth dictionary follows the same basic format, there are variations on the basic theme but all the elements remain conceptually the same, the specific implementation we will look at is W.Ragsdales fig-FORTH model published by the forth interest group in november 1980, it is available from the KAOS library for those who wish to read further.

As an example consider how the following would appear in memory

: TEST 0 DO TEST1 TEST2 LOOP ;

Each entry in the dictionary has four fields, these are :-

	address	bytes in hex	ascii	comments / field name.
NFA	5C8F	84		
	5C90	54 45 53 D4	TEST	
LFA	5C94	83 5C	LFA	link to NFA of last word
CFA	5C96	D9 07	CFA of :	
PFA	5C98	60 08	CFA of 0	
	5C9A	E1 03	CFA of (do)	
	5C9C	7F 5C	CFA of TEST1	
	5C9E	8B 5C	CFA of TEST2	
	5CA0	7A 03	CFA of (loop)	
	5CA2	FA FF	signed offset {back to (do)}	
	5CA4	0F 06	CFA of ;s	
	....	.....	....	

NAME FIELD ADDRESS (NFA)

This is the address of the header byte, the low 5 bits of the header byte contains the count of characters in the definition name, other bits of significance in the header byte are the preceedence bit and the smudge bit which we will discuss when we look at the forth compiler. The header byte is followed by the name itself, in ASCII with the high bit in the last character switched "on". This makes it easier to scan the dictionary structure when compiling.

LINK FIELD ADDRESS (LFA)

This location contains the address of the header byte of the last defined word in this vocabulary, as there may be multiple vocabularies co-existing within the one dictionary, each is seperately linked, but all ultimately linked to the trunk vocabulary FORTH (yes, you guessed it!, part 9 deals with vocabularies...).

CODE FIELD ADDRESS (CFA)

Now we start to get to the nuts and bolts, the CFA points to the CFA of another word.... HOWZAT, well, stay tuned and all will be revealed. In this

instance the word pointed to is the word DOCOL (mnemonic for DO COLON definition). Just by the way if we were looking at the definition of a variable or constant we would have found DOCON, or DOVAR, no prizes for guessing what the mnemonics are short for!. The CFA contents ALWAYS points to executable code. That is whatever is at the address pointed to by the CFA is executable machine code.

PARAMETER FIELD ADDRESS (PFA)

The parameter field consists of code field addresses of the words comprising the execution time behaviour of the word. Each word compiles only two (2) bytes into the dictionary. So that any forth word may be "called" by unsing only two bytes. In assembly language programs it takes T H R E E bytes to call a subroutine, this has lead some to claim that forth is more compact than assembly language. The techniques used to produce this compact means of extending the language revolve around the ADDRESS interpreter.

RELIGIOUS SEEKERS OF ULTIMATE TRUTH

This code is really quite simple if you make a mental change of gears and start thinking of the forth address interpreter as the fetch/execute cycle for a virtual machine, as we progress the concepts you are about to encounter will no doubt become revealed in all their elegant simplicity. Someone once said the simple concepts are the hardest to grasp, the forth concepts are indeed difficult to grasp at first. Once the pieces start to fall into place you will see why some FORTH-FANATICS go all misty eyed and wax philosophically about the beauty and power of the language.

THE FIRST PIECE OF THE PUZZLE. (DOCOL)

This can be viewed as a high (low?) level JSR, (subroutine call) this is where we first encounter the INSTRUCTION POINTER, and the CODE POINTER these pointers are in page zero for the 6502, but some processors have sufficient registers to utilize cpu registers.

```
generalized code        actual 6502 code

PUSH IP                 LDA $AF         push the instruction pointer on
                        PHA             the return stack, in this inst-
                        LDA $AE         ance the instruction pointer is
                        PHA             $AE,$AF
MOVE  W+2->IP           CLC             clear carry for addition to come
                        LDA $B1         here comes the code pointer!!
                        ADC #$02        step code pointer to the next
                        STA $AE         code field address in the word
                        TYA             being executed
                        ADC $B2         and store that as the new IP.
                        STA $AF
JMP NEXT                JMP NEXT        re-enter the execute cycle
```

What happens when you call a subroutine? well, first you save the return address, then you put the called address into the program counter. Now what is happening above is similiar to a subroutine call, the INSTRUCTION POINTER is identical in a lot of ways to the program counter. The code pointer $B1,$B2 is pointing to the address which contains the executable code. So how does it get executed you ask?. ..... enter "NEXT" stage left with fanfare.

PIECE TWO OF THE PUZZLE (NEXT)

Discovering NEXT is akin to striking oil....

```
generalized code          6502 code

MOVE (IP)->W              LDY #$01        load W with (IP) using post-
                          LDA ($AE),Y     indexed Y, note that $B1,$B2
                          STA $B2         are the target of an indirect
                          DEY             JMP at $00B0....
                          LDA ($AE),Y
                          STA $B1
IP+2->IP                  CLC             in this version of forth next
                          LDA $AE         increments the IP after the
                          ADC #$02        load indirect, some versions of
                          STA $AE         next are pre-incrementing
                          BCC GO.JUMP
                          INC $AF
JMP (W)        GO.JUMP    JMP $00B0       jump to an indirect jump....
                          .........
               $00B0      JMP (W)         at last we do it!!!
```

Some smart readers will note that all of this is really a double indirect jump, I.E. we might code it as :-

```
JMP  ((IP))+
```

Using motorola type assembler mnemonics to represent post-incrementing the register by the word size. Is there a microprocessor with such an instruction? The answer is YES.. the processor is the 68020, the 32 bit big brother to the 68000, the 68020 is due for release in U.S.A in sampling quantities later this year. So we can expect to see some spectacular performances running FORTH on this new machine. Incidentally it has a 32 bit data bus and runs at 16mHz. Light blue touch paper and stand well clear.

Back to the real world of the 6502...and cousins

THE RETURN TO CALLER (or go back from whence you came bit.)

This is what you do when you reach the end of a forth word and we have to have a way of returning to the calling word. The routine which performs this vital task is called SEMIS short for ;s SEMI-COLON S, you see how logical it is?. No? well read on..

```
generalized code          6502 code

POP IP                    PLA             remember we put the return address
                          STA $AE         on the stack way back in DOCOL.
                          PLA             .. so we just reverse the process
                          STA $AF
JMP NEXT                  JMP NEXT        re-enter and continue executing
                                          code as before
```

WHAT HAPPENS IF THE WORD CALLED ISN'T A COLON DEFINITION?

Remember that we jump to the contents of the contents of the CFA that address may point to ANY executable 6502 machine code. IF the word we are executing is a primitive code definition then the address contained at the CFA address will usually point to the PFA, when the routine terminates by simply jumping to NEXT the IP will step to the next address to be executed and in this way you may mix HI-LEVEL definitions with assembly language. The routine DOCOL is called by ALL hi-level colon definitions and all high level words terminate with SEMIS ;S, incidentally ";" compiles to ;S, wheras DOCOL is defined by the definition : with the (;CODE), discussion of defining words is to be covered at a later stage.

HOW TO FIND YOUR WAY AROUND THE DICTIONARY.

First some definitions that we are about to spring on you.

forth word	pronounced	stack effect	comments
'	"tick"	...--- PFA	find the PFA of the following word
CFA		PFA --- CFA	convert a PFA to a CFA
LFA		PFA --- LFA	convert a PFA to a LFA
NFA		PFA --- NFA	convert a PFA to a NFA
PFA		NFA --- PFA	convert a NFA to a PFA
LATEST		... --- NFA	leave address of NFA of topmost word in CURRENT vocabulary
ID.		NFA --- ...	print the name of the word

Lets write a word to list the entire contents of the forth dictionary.

```
: WORDS  LATEST                ( get the topmost NFA in this vocabulary )
         BASE @ >R             ( save the I/O base on the return stack  )
    BEGIN
          DUP CR ID.           ( print the name of the word             )
          PFA LFA @            ( get the nfa of next word down          )
          DUP 0=               ( test to see if link is zero            )
          ?TERMINAL            ( see if operator wants to stop listing  )
          OR                   ( combine the flags, either break or end )
   UNTIL                       ( conditional branch back to BEGINning   )
        R> BASE ! ;            ( restore I/O base back to what it was   )
```

The saving and restoring of the I/O base isn't really required in the above definition, however, it has been left that way so that you can modify the definition to produce whatever sort of fancy listing you desire. For example you might want to build a decompiler into the dictionary listing, or list all the relevent addresses of the various words.

Another example, suppose you want to examine some part of the dictionary to see what was compiled :-

```
' EXAMPLE NFA 30 DUMP
```

This will find the word using "tick" and then convert the PFA found into the name field address, then dump 30 bytes from that address.

DECOMPILING

Decompilers are easy to write in forth, but beware of words with CFA's that don't point to known structures.

ADVANCED READERS ONLY.

If a new defining word is added to the dictionary then it is neccessary to inform the decompiler as to how the new structure is to be identified and decompiled, for example embedded strings and data structures may be built using a <BUILDS DOES> construction or some I/O words have CFA's which point directly to routines in DOS or a rom based monitor. All of these cases can be handled by a case statement within the decompiler to identify what is being decompiled.

SUMMARY AND CONCLUSIONS

The forth address interpreter can be visualised as a virtual machine with the COLON definition representing a list of subroutine calls, the microcode for these "subroutine calls" are the routines DOCOL = JSR and SEMIS = RTS.

The code is threaded indirectly, making the NEXT routine a double indirect jump to the contents of the contents of the Instruction pointer.

ASSEMBLER versus DIRECT THREADED

Imagine you are writing a program in assembly language and having structured the program and written all the I/O drivers and low level subroutines. How does DIRECT threaded code compare. Well the 6502 doesn't have an indirect indexed jump instruction, but the W65SC816 does....

Assembler	bytes	Direct threaded code		bytes
JSR ROUTINE1	3	BASE	ADDRESS1	2
JSR ROUTINE2	3		ADDRESS2	2
JSR ROUTINE3	3		ADDRESS3	2
.....			.....	
ROUTINE1 .....RTS			ROUTINE1....X+2->X,JMP (BASE),X	
ROUTINE2 .....RTS			ROUTINE2....X+2->X,JMP (BASE),X	
ROUTINE3 .....RTS			ROUTINE3....X+2->X,JMP (BASE),X	

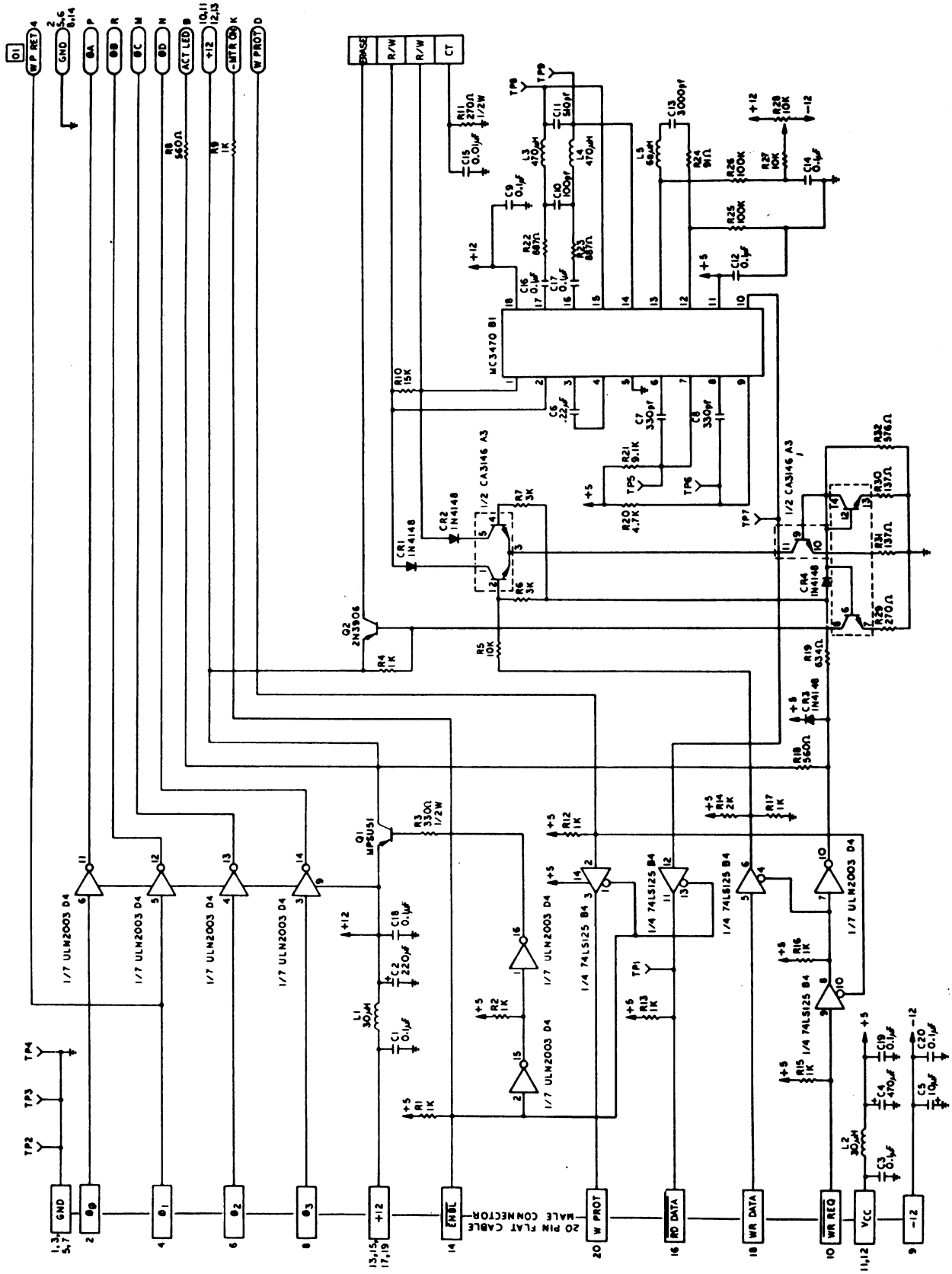
Each subroutine call requires 12 clock cycles 6 for the JSR and 6 for the RTS whereas the INX,INX, JMP (HHHH),X sequence requires only 10 clock cycles and doesn't use any stack space, so the stack can be used for parameter passing with relative ease, further each subroutine call takes 3 bytes of memory wheras the threaded code requires only 2 bytes for each subroutine call, however each routine terminates with 5 bytes rather than 1. So memory saving depends on how many subroutines there are compared with the number of subroutine calls. Being able to easily pass parameters on the stack could well save considerable code as well as reduced program size.

****

What we are describing is an INTERPRETER that is FASTER and more compact than the way most programmers normally write assembly language!

****

Now what is required is a fast 32 bit processor that can handle the indirect threaded code .....Hmmmm...68020 anybody?



Registered by Australia Post
Publication No. VBG4212

If undeliverable return to
KAOS, 10 Forbes St
Essendon, Victoria 3040

K A O S K A O S
K Postage K
A Paid A
O Essendon O
S 3040 S
K A O S K A O S

KAOS

THE VICTORIAN 6502 USER GROUP

For People Who Have Got Smart

10 Forbes St.
Essendon
Victoria 3040
Australia
Ph. 375 3478

OSI SYM KIM AIM UK101 RABBLE65 BBC APPLE

SURNAME_____ CHRISTIAN NAME_____

ADDRESS_____

_____ POST CODE_____

COUNTRY_____ PHONE_____

FEES TO JANUARY 1986

$15 Australia, $20 N.Z. & P.N.G., $25 Europe & U.S.A.

My name and address may/may not be included in lists circulated to other KAOS members.

TYPE OF SYSTEM_____

TERMINAL WIDTH? 24 32 48 64 80

HOW MUCH MEMORY?_____

DISK DRIVES?_____ 1 or 2?_____

WHAT SIZE? 3½" 5¼" 8"

CASSETTE?_____

PRINTER?_____

MODEM?_____

ARE YOU A MEMBER OF A NETWORK?_____

USER NAME?_____

AMATEUR RADIO?_____ CALL SIGN_____ RTTY_____

INTERESTS?_____

_____

OSI SYM KIM AIM UK101 RABBLE65 BBC APPLE